Algorithms and combinatorics for geometric graphs

Lecture 3

Efficient algorithms on planar graphs

september 28, 2023

Luca Castelli Aleardi
Graph separators
Divide & Conquer for (planar) graphs: *Small Separators*

Tool for recursive decompositions of graphs

Many Algorithmic applications:
- Approximation scheme for *Maximum Independent Set*
- Graph Encoding: *compression schemes* and *compact representations*
- Graph Drawing: *spherical parameterizations*
- Point Location (in optimal time)
Divide & Conquer for (planar) graphs: Small Separators

Encoding planar graphs in $O(n)$ bits

Approximation scheme for Maximum Independent Set

Graph Drawing: spherical parameterizations

Compute 2D layouts of each hemisphere

Tutte barycentric layout (in 2D)
Graph separators: definition
Given a graph $G = (V, E)$ with $n$ vertices, an $\varepsilon$-separator is a partition $(A, B, S)$ of the vertices such that:

- (balanced) every connected component of $G \setminus S$ has size at most $\varepsilon n$
- (separation) there are no edges between $A$ and $B$
- $S$ is small: $|S| = o(n)$
Separators: definitions

**Def**

Given a weighted graph $G = (V, E)$ with $n$ vertices and total weight $W$, a separator is a partition $(A, B, S)$ of the vertices such that:

- (balance) every connected component of $G \setminus S$ has weight at most $\frac{1}{2}W$
- (separation) there are no edges between $A$ and $B$
- $S$ is small: $|S| = O(\sqrt{n})$
Separators for trees

**Lemma:** A weighted tree $T$ admits a separator consisting of a single vertex (computable in $O(n)$ time)

**Proof:**

**First step:** compute for each vertex $v \in T$ the weight of the subtree $t_v$ rooted at $v$ (total overall cost: linear time)

$$T \setminus v := C_1(v) \cup C_2(v) \cup \ldots$$

Case 1: $W(C_i(v)) \leq \frac{1}{2} W$ $\forall i$

return $v$

Case 2: $W(C_{\text{max}}(v)) > \frac{1}{2} W$

move to the descendant $w \in C_{\text{max}}(v)$

restart from $w$

**Correctness:** The algorithm visit each vertex at most once (we move from $v$ to its descendant $w$) The component $C_i(w)$ containing $v$ is small: $W(C_i(w)) \leq W - W(C_{\text{max}}(v)) \leq \frac{1}{2}$
Separators: definitions

examples: what about grid graphs? and general graphs? planar graphs?
Planar Separators theorems

Thm (Lipton-Tarjan, ’79)

Every planar graph with $n$ vertices admits a $\frac{2}{3}$-separator of size at most $4\sqrt{n}$, that can be computed in linear time.

(purely combinatorial proof: perform a BFS traversal)

Thm (Spielman and Teng)

Every planar graph with $n$ vertices admits a $\frac{3}{4}$-separator of size (in expectation) at most $2\sqrt{n}$.

(geometric proof: omitted)

sphere packing (Koebe)

stereographic projection + Möbius transformation

Compute intersections with a random hyperplane passing through the origin
Let $G$ be a planar weighted graph with $n$ vertices. Let $U$ be a BFS spanning tree of $T$ of depth at most $d$, rooted at $r$. Then we can compute in linear time a separator of size at most $3d + 1$.

**Proof (assume the graph is triangulated)**

Construct a weighted dual graph $G^*$: each face (a dual vertex) get the weight of a vertex in $G$ and each vertex assigns its weight to a unique incident face.

Define the spanning tree $T^* := G^* \setminus U^*$

Apply previous Lemma to $T^*$, getting a separating vertex $c^*$ (all component of $T^* \setminus c^*$ are small, of cost at most $\frac{1}{2}$)

computes three shortest paths $P_i(t)$ from $t$ to the root vertex $r$

$$S := t \cup P_1 \cup P_2 \cup P_3$$

**Claim 1:** The separator $S$ has at most $3d + 1$ vertices

**Claim 2:** Each component $C$ of $G \setminus S$ has weight at most $\frac{1}{2}$

since each component $C^*$ of $T^* \setminus c^*$ has weight at most $\frac{1}{2}$

and the total (inner) weight of $C$ is at most the weight of $C^*$
Planar Separators for graphs of small radius

Theorem
Let $G$ be a connected planar graph with $n$ vertices. Then we can compute in linear time a separator of size at most $O(\sqrt{n})$.

Proof: Compute a BFS spanning tree $T$ of $G$, rooted at $r$

Claim 1:
The set of vertices $L_i$ at level $l_i$ are a separator (splitting $G$)

- Define $l_m :=$ median level
  \[ \sum_{i < m} W(L_i) \leq \frac{1}{2} \]
  \[ \sum_{i > m} W(L_i) \leq \frac{1}{2} \]

- Define $l_{\inf} :=$ largest level $l_j$ ($j < i$) such that $|L_{l_{\inf}}| \leq \sqrt{n}$
- Define $l_{\sup} :=$ smallest level $l_j$ ($j > i$) such that $|L_{l_{\sup}}| \leq \sqrt{n}$

Remark: The levels $l_k$ between $l_{\inf}$ and $l_{\sup}$ are large: $|L_k| \geq \sqrt{n} + 1$ (for $\inf < k < \sup$)

Claims:
- Number of levels $l_k$ between $l_{\inf}$ and $l_{\sup}$: $l_{\sup} - l_{\inf} \leq \frac{n}{\sqrt{n} + 1} < \sqrt{n}$
- The set of vertices $S' := L_{l_{\inf}} \cup L_{l_{\sup}}$ is small: $|S'| \leq 2\sqrt{n}$
- The connected components of $G \setminus S'$ which are large (weight larger than $\frac{1}{2}$) are between the levels $l_{\inf}$ and $l_{\sup}$ (by definition $l_m :=$ median level)
Planar Separators for graphs of small radius

Lemma

Let $G$ be a connected planar graph with $n$ vertices. Then we can compute in linear time a separator of size at most $O(\sqrt{n})$.

Proof: Compute a BFS spanning tree $T$ of $G$, rooted at $r$

Claim 1: The set of vertices $L_i$ at level $l_i$ are a separator (splitting $G$)

\[
\text{define } l_m := \text{median level} \quad \sum_{i<m} W(L_i) \leq \frac{1}{2} \\
\sum_{i>m} W(L_i) \leq \frac{1}{2}
\]

Last step:

Take the graph $G'$ induced by the vertices strictly between the levels $l_{\text{inf}}$ and $l_{\text{sup}}$

$G'$ is not necessarily connected: create a graph $G''$ by adding a dummy vertex $r'$ and connecting it to vertices in $l_{\text{inf}}$

Apply previous Lemma to graph $G'$: its radius is $O(\sqrt{n})$, so the separator $S$ has size $O(\sqrt{n})$

\text{return } L_{l_{\text{inf}}} \cup L_{l_{\text{sup}}} \cup S
Graph separators: algorithmic applications
(classical) Graph representations

**Adjacency matrix**

\[
A_G[i, j] = \begin{cases} 
1 & \text{if } v_i \text{ adjacent } v_j \\
0 & \text{otherwise}
\end{cases}
\]

\[
\begin{bmatrix}
0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 \\
\end{bmatrix}
\]

\(O(n^2)\) bits

**Adjacency list (and its variants)**

<table>
<thead>
<tr>
<th>(d_i)</th>
<th>(O(n \log n)) bits</th>
<th>(O(n \log n)) bits</th>
<th>(O(n \log n)) bits</th>
<th>(O(n \log n)) bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v_1)</td>
<td>3 2 3 4</td>
<td>3 2 3 4</td>
<td>3 1 1 1</td>
<td>3 1 1 1 1</td>
</tr>
<tr>
<td>(v_2)</td>
<td>4 1 4 5 3</td>
<td>4 1 3 4 5</td>
<td>4 –1 2 1 1</td>
<td>4 0 1 2 1 1</td>
</tr>
<tr>
<td>(v_3)</td>
<td>4 5 4 1 2</td>
<td>4 1 2 4 5</td>
<td>4 –2 1 2 1</td>
<td>4 0 2 1 2 1</td>
</tr>
</tbody>
</table>

neighbors in arbitrary order

sorted neighbors

difference encoding

difference encoding

negative differences

positive differences

sign
Encoding of planar graphs in $O(n)$ bits

**Thm**

Any planar graph with $n$ vertices can be encoded with at most $O(n)$ bits.

**Solution:** use difference encoding of adjacency lists + separators

this time we get $O(n)$ bits

**Why does it work?** Because vertices which are ”close” in the graph get ”close indices”

<table>
<thead>
<tr>
<th>$d_i$</th>
<th>sign</th>
<th>positive differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1 2 1 1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2 1 2 1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Any planar graph with \( n \) vertices can be encoded with at most \( O(n) \) bits.

**Proof (overview):**

1. **Step 1:** compute a recursive decomposition using (edge) separators

   \[
   |G| := n \\
   |S| = O(\sqrt{n}) \\
   |G_1| \leq O(\alpha n) \\
   |G_2| \leq O(\alpha n)
   \]

2. **Step 2:** encode using adjacency lists with difference encoding

   - encode the edges in \( S \) as usual
     \[
     \text{size}(S) = O(|S| \log |S|)
     \]
     \[
     \text{size}(S) = O(\sqrt{|G|} \log |G|)
     \]
   - encode each piece \( G_i \) recursively
     \[
     \text{size}(G) = \text{size}(S) + \text{size}(G_1) + \text{size}(G_2)
     \]
     \[
     \text{size}(n) = C \cdot \sqrt{n \log n} + \text{size}(\alpha n) + \text{size}(\alpha n)
     \]
   - \( \text{size}(n) = O(n) \)
Recursive graph decompositions and hierarchical representations

**Thm (Lipton Tarjan)**

Given a planar graph $G$ of size $n$ and weight $W = 1$, and a parameter $0 \leq \varepsilon \leq 1$. Then it is possible to compute a separator $S \subset V$ of size at most $|S| = O\left(\frac{n}{\varepsilon}\right)$, such that each connected component of $G \setminus S$ has size at most $\varepsilon$. The computation time is $O(n \log n)$.
Maximum Independent Set

Thm (approx scheme)
Let $G$ be a planar graph on $n$ vertices. Show that you can compute in $O(n \log n)$ time an approximated independent set of vertices $I$ whose size, for large values of $n$, is closed to the size of a maximum independent set $I_{\text{opt}}$: 
$$\left| \frac{|I| - |I_{\text{opt}}|}{|I_{\text{opt}}|} \right|$$ tends to 0 with increasing $n$.

Proof:
Let \( G \) be a planar graph on \( n \) vertices. Show that you can compute in \( O(n \log n) \) time an approximated independent set of vertices \( I \) whose size, for large values of \( n \), is closed to the size of a maximum independent set \( I_{\text{opt}} \):

\[
\left| I \right| - \left| I_{\text{opt}} \right| \leq O\left(\frac{n}{\sqrt{\log \log n}}\right)
\]

with increasing \( n \).

**Proof:**

Use uniform weights: \( w(v_i) = \frac{1}{n} \).

**Idea:** apply previous result with parameter \( \varepsilon = \frac{\log \log n}{n} \).

- Sub-components \( G_i \) have size \( |G_i| \leq \frac{W(G_i)}{n} = O(\log \log n) \).
- The vertex separator \( S \) has size at most \( |S| = O\left(\frac{n}{\sqrt{\log \log n}}\right) \).

**Trick:** in each \( G_i \) use **brute-force** to compute a maximal independent set (checking all subsets) for each \( G_i \) of size \( n_i \) it takes: \( O(n_i \cdot 2^{n_i}) \) in overall: \( O\left(\frac{n}{\log \log n} (\log \log n) \cdot 2^{\log \log n}\right) = O(n \log n) \).

**Remark:** planar graphs are 4-colorable

\[
\frac{|I| - |I_{\text{opt}}|}{|I_{\text{opt}}|} \leq O\left(\frac{n/\sqrt{\log \log n}}{n/4}\right) = O\left(\frac{1}{\sqrt{\log \log n}}\right)
\]
Computing triangles and cliques in planar graphs
Let $G$ be a graph on $n$ vertices and $m$ edges. Then it is possible to count (or list) the triangles of $G$ in $O(nm)$ time.

**Thm**

Let $G$ be a graph on $n$ vertices and $m$ edges. Then it is possible to count (or list) the triangles of $G$ in $O(nm)$ time.

**Proof:**

```plaintext
procedure COUNT_TRIANGLES(G = (V, E))
    Count := 0;
    for each vertex $u \in V$
        mark all vertices which are neighbors of $u$ in $G$;
        for each marked vertex $v \in V$
            do { for each vertex $w$ which is a neighbor of $v$ in $G$
                    do if $w$ is marked then Count := Count + 1;
                    unmark vertex $v$;
                    $G := G \backslash \{u\}$; // vertex removal in $O(d_u)$ time
            }
    return Count;
```

triangle:= cycle of size 3 (complete graph on 3 vertices)
### Counting triangles

**Thm**

Let $G$ be a graph on $n$ vertices and $m$ edges. Then it is possible to count (or list) the triangles of $G$ in $O(nm)$ time.

**Proof:**

```plaintext
procedure COUNT_TRIANGLES(G = (V, E))
    Count := 0;
    for each vertex $u \in V$
        mark all vertices which are neighbors of $u$ in $G$;
        for each marked vertex $v \in V$
            do 
                for each vertex $w$ which is a neighbor of $v$ in $G$
                    do if $w$ is marked then Count := Count + 1;
                unmark vertex $v$;
            $G := G \setminus \{u\}$; // vertex removal in $O(d_u)$ time
    return Count;
```

- each vertex $v$ is marked at most $\text{deg}(v)$ times: each time the inner loop performs at most $\text{deg}(v)$ iterations: the cost per vertex is thus at most $\text{deg}(v)^2$
- $\sum_{v \in V} \text{deg}^2(v) \leq (\max_{v \in V} \text{deg}(v)) \cdot (\sum_{v \in V} \text{deg}(v)) \leq (|V| - 1) \sum_{v \in V} \text{deg}(v) = O(|V||E|)$
Let $G$ be a planar graph on $n$ vertices and $m$ edges. Then it is possible to count (or list) the triangles of $G$ in $O(n)$ time.

**Thm**

Let $G$ be a planar graph on $n$ vertices and $m$ edges. Then it is possible to count (or list) the triangles of $G$ in $O(n)$ time.

**Proof:**

**procedure** COUNT_TRIANGLES($G = (V,E)$)

$Count := 0$; order vertices of $V$ according to non-increasing degree as $(u_1, \ldots, u_n)$

for each vertex $u \in V$ // visit vertices according to computed order
  
  mark all vertices which are neighbors of $u$ in $G$;

  for each marked vertex $v \in V$
    do
      for each vertex $w$ which is a neighbor of $v$ in $G$
        do if $w$ is marked then $Count := Count + 1$;

    unmark vertex $v$;

  $G := G \setminus \{u\}$; // vertex removal in $O(d_u)$ time

return $Count$;

- for any edge $\{u, v\}$ for a pair of vertices $u, v$ considered in the algorithm, we have $\deg(v) \leq \deg(u)$
- the time complexity becomes $\sum_{(u,v)\in E} \min(d_u, d_v)$

**Claim (exercise, homework I)**

Show that in a planar graph with $n$ vertices we have:

$$\sum_{(u,v)\in E} \min\{\deg(u), \deg(v)\} \leq 18n$$
Let $G$ be a planar graph on $n$ vertices and $m$ edges. Then it is possible to count (or list) all 4-cliques of $G$ in $O(n)$ time.

**Thm**

Let $G$ be a planar graph on $n$ vertices and $m$ edges. Then it is possible to count (or list) all 4-cliques of $G$ in $O(n)$ time.

**Proof:** [case analysis, exercise]

**Claim 1:**

- Consider a 4-clique $Q = \{u, v, w, x\}$ in $G$.

  Show that the four vertices $u, v, w, x$ cannot all belong to the same level $V_j$.

**Claim 2:** consider a 4-clique $Q = \{u, v, w, x\}$ in $G$, and let $j$ be a positive integer $\leq k$.

- assume $u \in V_{j-1}$ and $v, w, x \in V_j$. Show that for one of the tree vertices $v, w, x$ the only incident edge lying in $E_j$ has $u$ has other extremity.

- assume $u, w, x \in V_{j-1}$ and $x \in V_j$. Show that the edges incident to $x$ lying in $E_j$ are exactly $(u, x), (v, x)$ and $(w, x)$.

- assume $u, v \in V_{j-1}$ and $w, x \in V_j$. Show that one of the vertices $w, x$ has exactly two incident edges lying in $E_j$ (whose other extremities are $u$ and $v$).

**Hint:** compute a BFS of $G$ and partition the vertices into $k + 1$ sets $\{V_0, V_1, \ldots, V_k\}$

- $V_k :=$ vertices at the distance $k$ from the root (seed) vertex

- define $E_j :=$ set of edges $e = (u, v)$ s. t. $u \in V_{j-1}$ and $v \in V_j$

  (an edge belongs to $E_j$ if it is connecting two vertices on levels $V_j$ and $V_{j-1}$).
Let $G$ be a planar graph on $n$ vertices and $m$ edges. Then it is possible to count (or list) all 4-cliques of $G$ in $O(n)$ time.

**Thm**

Let $G$ be a planar graph on $n$ vertices and $m$ edges. Then it is possible to count (or list) all 4-cliques of $G$ in $O(n)$ time.

**Proof:** [case analysis, exercise]

**Hint:** compute a BFS of $G$ and partition the vertices into $k + 1$ sets $\{V_0, V_1, \ldots, V_k\}$

- $V_k :=$ vertices at the distance $k$ from the root (seed) vertex
- define $E_j :=$ set of edges $e = (u, v)$ s. t. $u \in V_{j-1}$ and $v \in V_j$

(an edge belongs to $E_j$ if it is connecting two vertices on levels $V_j$ and $V_{j-1}$)

**Claim 1:**

- Consider a 4-clique $Q = \{u, v, w, x\}$ in $G$.
  Show that the four vertices $u, v, w, x$ cannot all belong to the same level $V_j$.

**Claim 1:**

Contracting the edges we get $K_5$.

\[ u \quad v \quad w \quad s \quad x \]